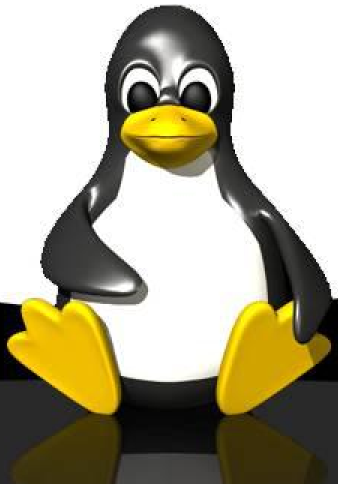
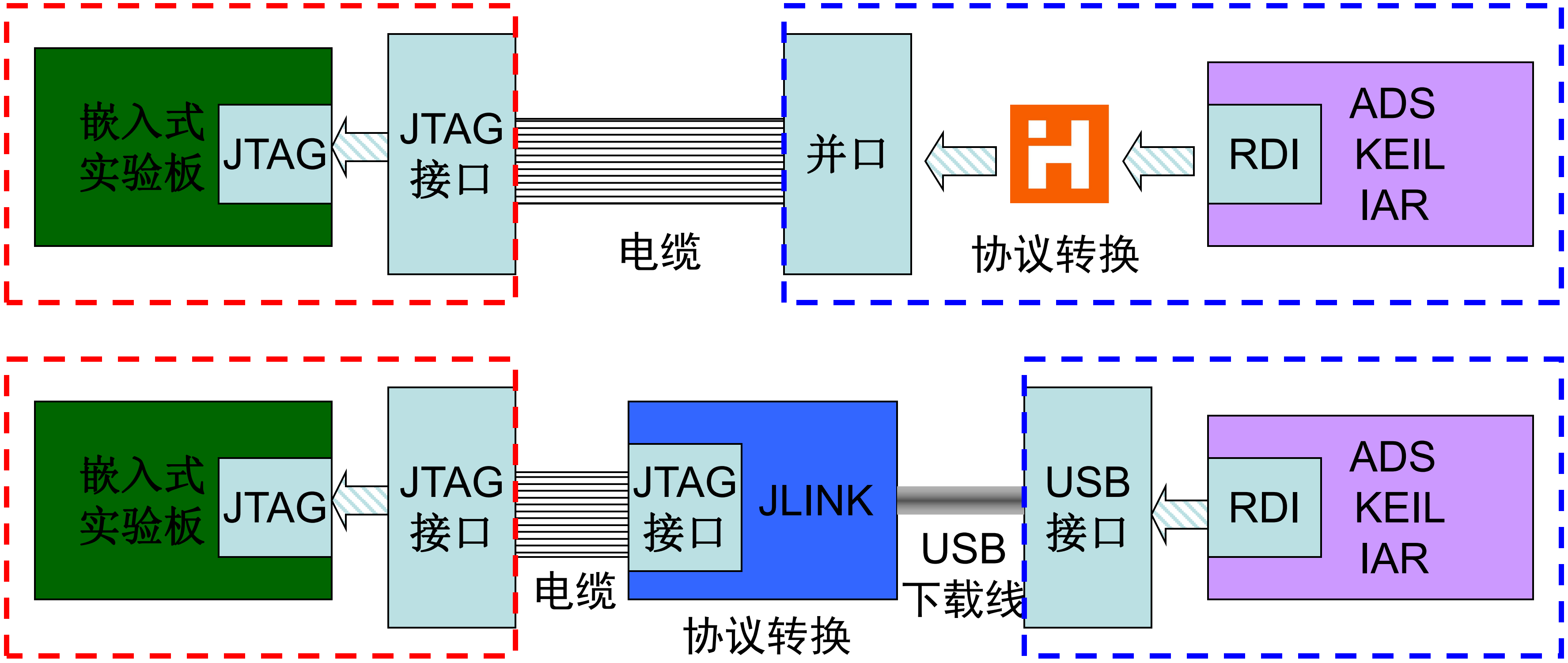
JLINK调试仿真及烧写FLASH教程

J—LINK ？

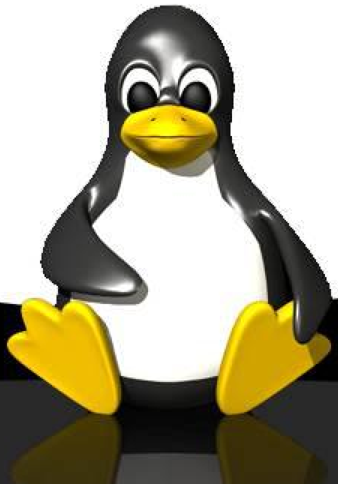


This is j-link



JTAG：国际标准测试协议
RDI：ARM公司提出的调试接口标准

为什么要使用JLINK?



1、**JLINK**用硬件进行协议转换，烧写、仿真速度快。
2、支持的芯片多。
3、**JLINK**使用**USB**下载线与计算机相连，仿真、烧写程序非常方便。

Part 1

JLINK+AXD+S3C4510B调试仿真

A、配置ADS工程 ※

B、编译并进入AXD

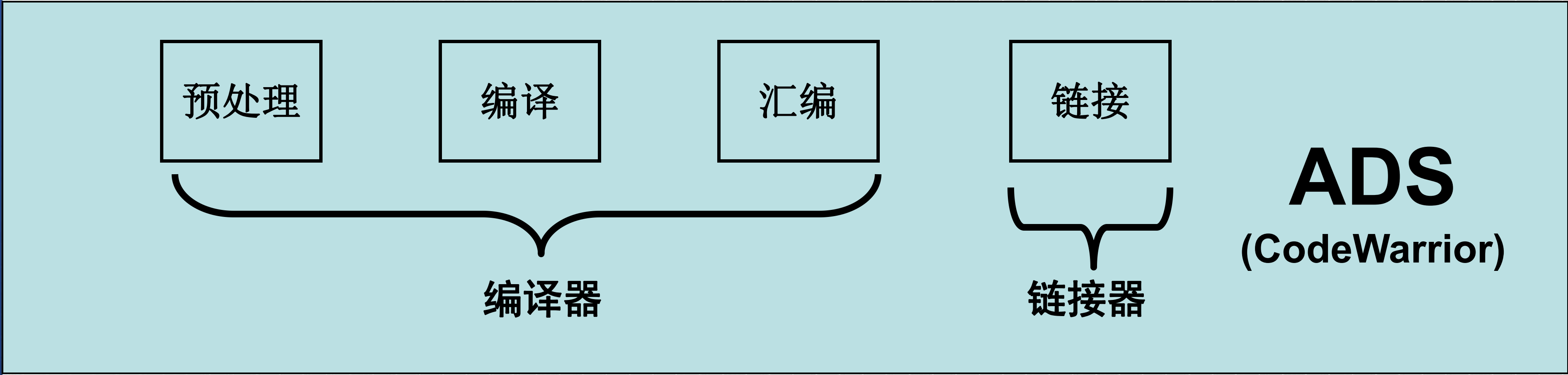
C、配置AXD ※

D、调试仿真

配置ADS工程

菜单>Edit>DebugRel Settings

高级语言程序从源代码到成为可在硬件上运行的可执行代码需要经历四个阶段：



因此，我们配置**ADS**工程主要配置编译器和链接器。

配置ADS工程

菜单>Edit>DebugRel Settings

主要配置的选项：

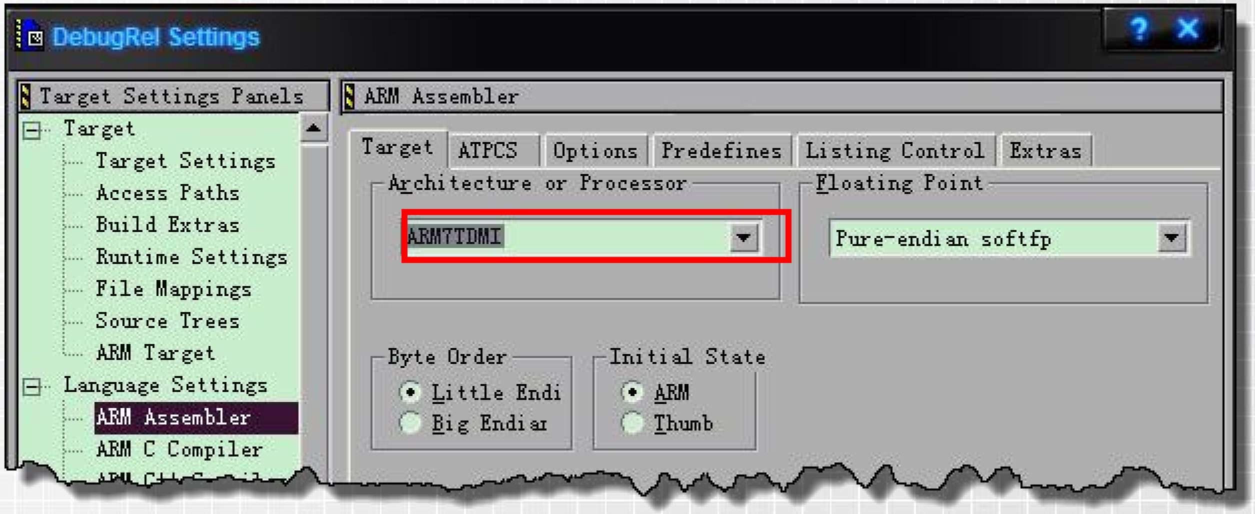
1、ARM Assembler（编译器）
2、ARM C Compiler（编译器）
3、ARM Linker（链接器）

其他选项默认即可。

配置ADS工程

菜单>Edit>DebugRel Settings

1、ARM Assembler



DebugRel Settings

Target Settings Panels

- Target
  - Target Settings
  - Access Paths
  - Build Extras
  - Runtime Settings
  - File Mappings
  - Source Trees
  - ARM Target
- Language Settings
  - ARM Assembler
  - ARM C Compiler

ARM Assembler

Target | ATPCS | Options | Predefines | Listing Control | Extras

Architecture or Processor: ARM7TDMI

Floating Point: Pure-endian softfp

Byte Order: [✓ Little Endi] Big Endiar

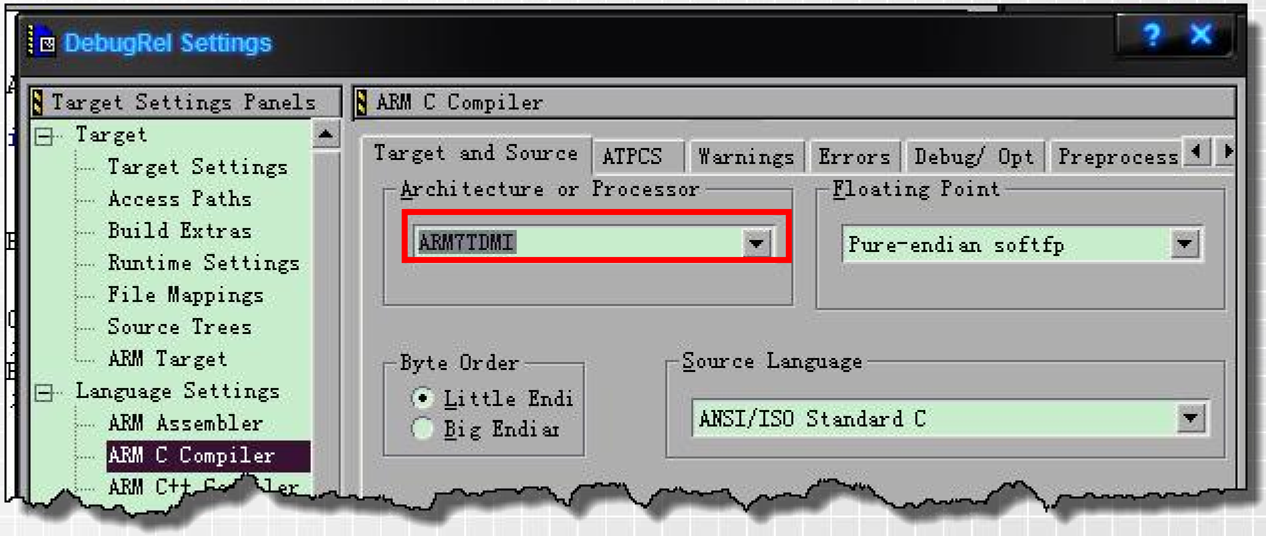
Initial State: [✓ ARM] Thumb

目的：选择与ARM核相匹配的汇编代码编译器

配置ADS工程

菜单>Edit>DebugRel Settings

2、ARM C Compiler



ARM C Compiler

Architecture or Processor: ARM7TDMI

Floating Point: Pure-endian softfp

Byte Order: [✓ Little Endi] Big Endian

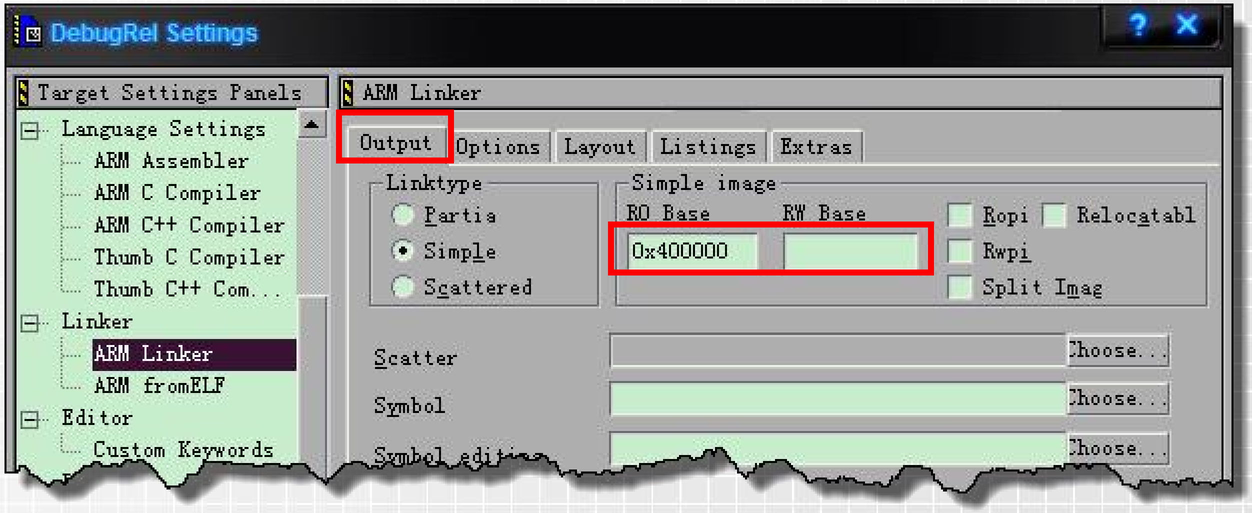
Source Language: ANSI/ISO Standard C

目的：选择与ARM核相匹配的C代码编译器

配置ADS工程

菜单>Edit>DebugRel Settings

3、ARM Linker（**Output**）



Linktype: Partia, [✓ Simple], Scattered

Simple image: RO Base 0x400000, RW Base

Ropi, Relocatabl, Rwpi, Split Imag

Scatter

Symbol

Symbol editing

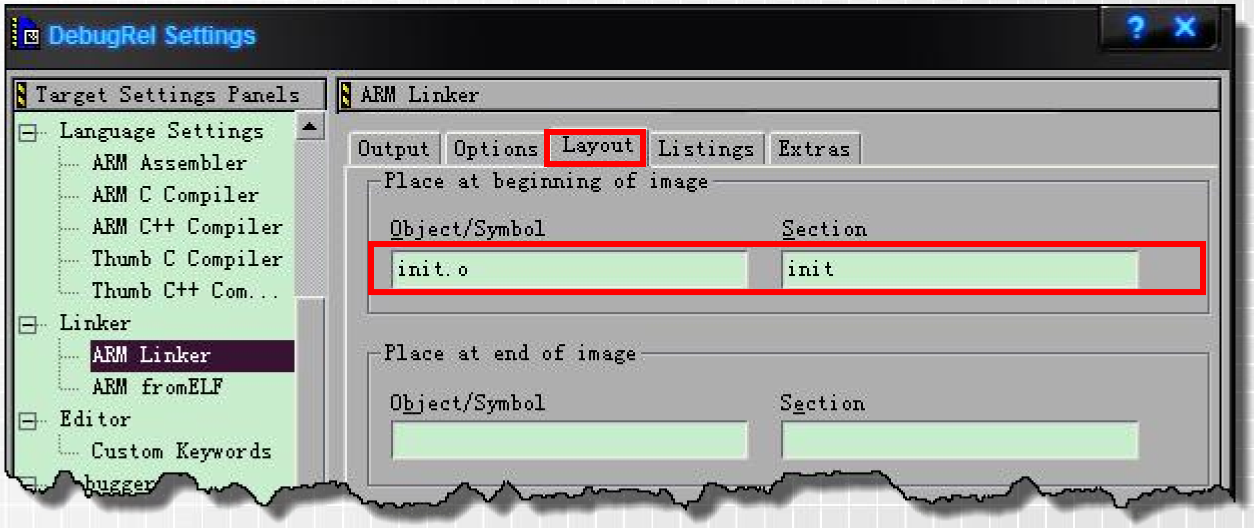
目的：设置代码段的起始地址为RAM的起始地址。

在ARM的集成开发环境中，只读的代码段和常量被称作RO段(ReadOnly)；可读写的全局变量和静态变量被称作RW段(ReadWrite)；RW段中要被初始化为零的变量被称为ZI段(ZeroInit)。

配置ADS工程

菜单>Edit>DebugRel Settings

3、ARM Linker（**Layout**）



Place at beginning of image

Object/Symbol	Section
init.o	init

Place at end of image

Object/Symbol	Section

目的：

1、指定放置在可执行文件开头的目标文件为init.o

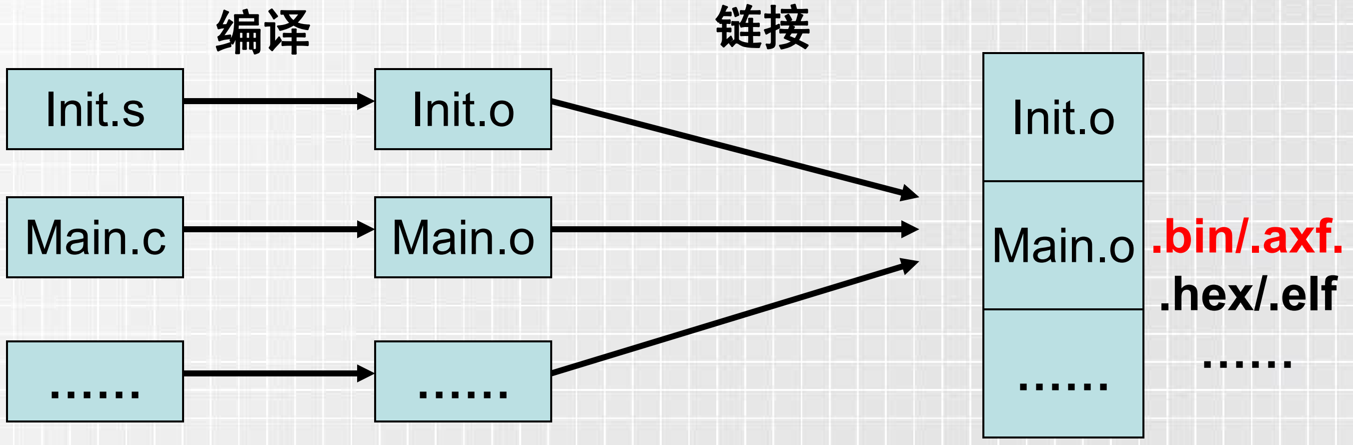
2、指定放置的逻辑段的段名为init（本例中init为代码段）

配置ADS工程

菜单>Edit>DebugRel Settings

init.o？

汇编结束后，生成多个目标文件，一般是一个源代码文件生成一个目标文件（头文件除外），然后由链接器来把这些目标文件链接成一个可执行的二进制代码文件。这个文件可用来调试或者烧写到ROM中。



Init？

```
CODE32
AREA    Init,CODE,READONLY
ENTRY                        ;指定程序入口地址
```

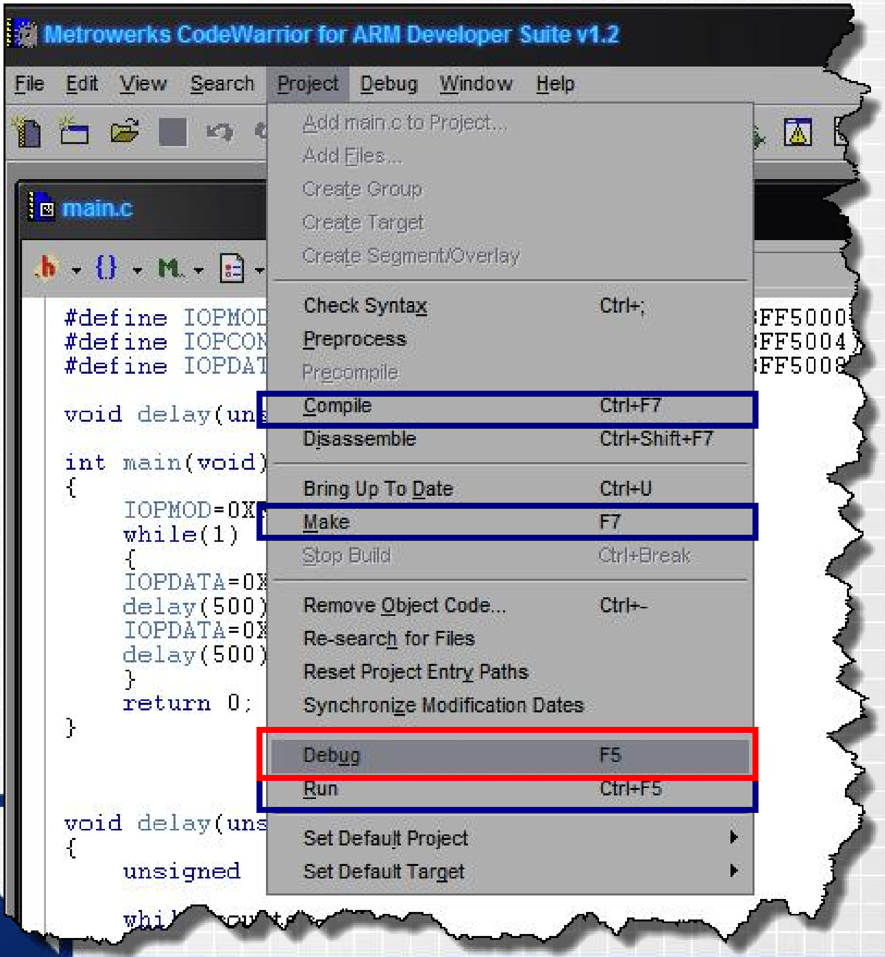
配置ADS工程

菜单>Edit>DebugRel Settings

配置完毕之后一定要注意存盘（ctrl+s），这样配置才会生效，否则编译时会报错。

编译并进入AXD

菜单>Projiect>Debug



```
#define IOPMOD
#define IOPCON
#define IOPDAT

void delay(un

int main(void)
{
    IOPMOD=0X
    while(1)
    {
    IOPDATA=0X
    delay(500)
    IOPDATA=0X
    delay(500)
    }
    return 0;
}


void delay(uns
{
    unsigned

    whi
```

命令介绍

Compile：
编译单个源文件,生成一个.o文件。

Make：
编译整个工程，生成多个.o文件和一个.axf文件。

Debug：
编译整个工程，生成多个.o文件和一个.axf文件，并进入AXD。

Run：
编译整个工程，生成多个.o文件和一个.axf文件，并进入AXD，同时运行程序。

配置AXD

AXD的配置步骤只需以下两步：

一、加载初始化指令

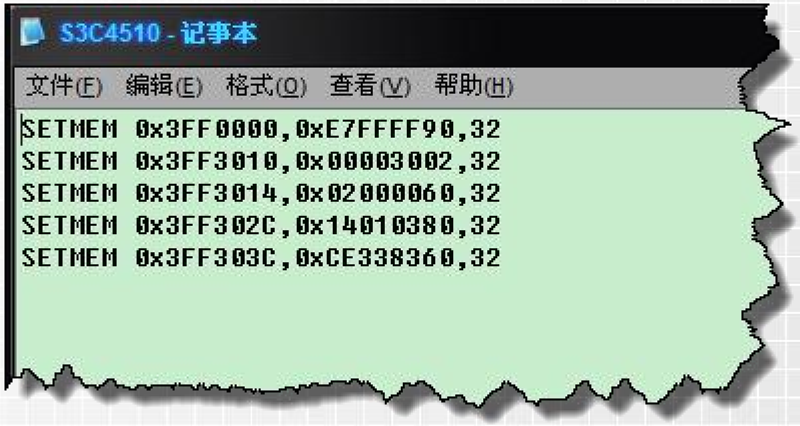
二、加载**JLINK**动态链接库

配置AXD

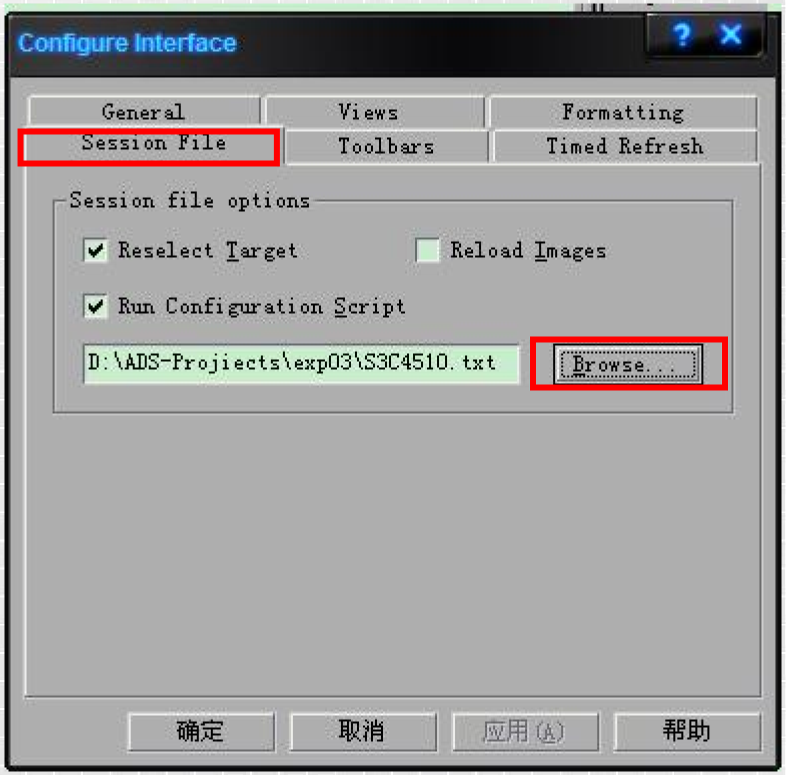
一、加载初始化指令

1、新建一个文本文档，输入下图所示的初始化指令，然后保存以备用。

2、菜单>Options>Configure Interface，加载刚才的初始化文本。



```
SETMEM 0x3FF0000,0xE7FFFF90,32
SETMEM 0x3FF3010,0x00003002,32
SETMEM 0x3FF3014,0x02000060,32
SETMEM 0x3FF302C,0x14010380,32
SETMEM 0x3FF303C,0xCE338360,32
```



Session file options

✓ Reselect Target　　☐ Reload Images

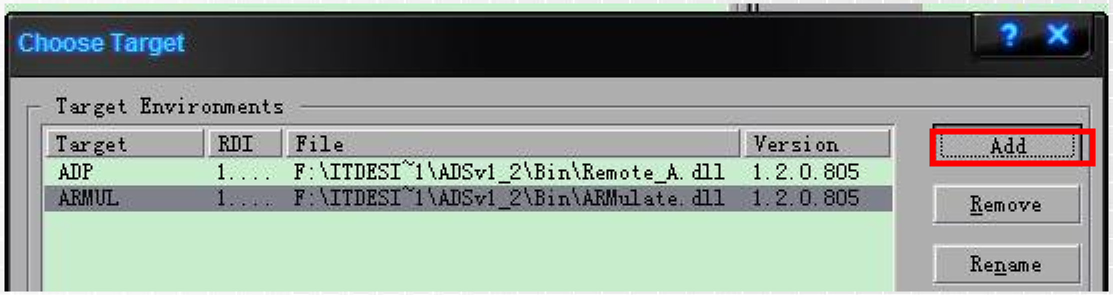
✓ Run Configuration Script

D:\ADS-Projiects\exp03\S3C4510.txt

配置AXD

二、加载JLINK动态链接库

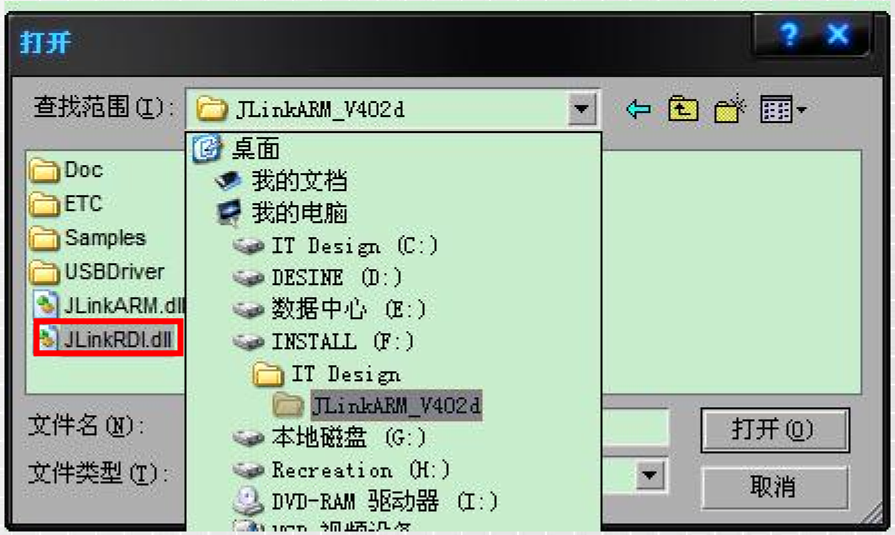
菜单>Options> Target，加载动态链接库JLinkRDI.dll，该文件在J-link软件的安装目录下。



Choose Target

Target Environments

Target	RDI	File	Version
ADP	1....	F:\ITDESI~1\ADSv1_2\Bin\Remote_A.dll	1.2.0.805
ARMUL	1....	F:\ITDESI~1\ADSv1_2\Bin\ARMulate.dll	1.2.0.805



打开

查找范围(I): JLinkARM_V402d

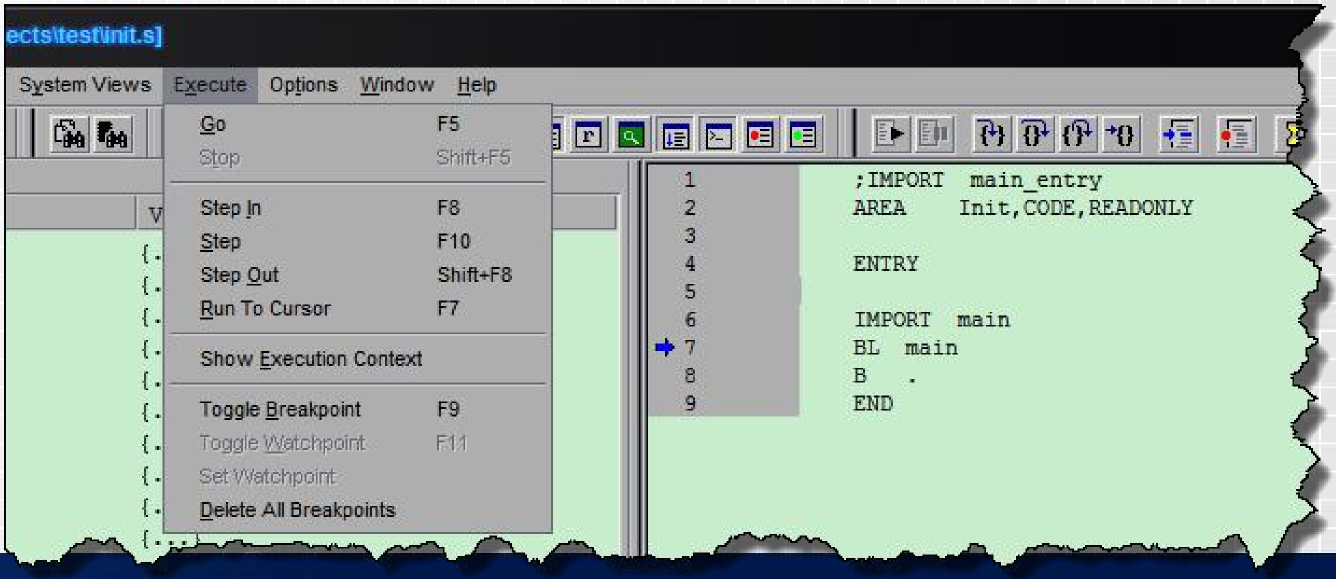
Doc
ETC
Samples
USBDriver
JLinkARM.dll
JLinkRDI.dll

桌面
我的文档
我的电脑
IT Design (C:)
DESINE (D:)
数据中心 (E:)
INSTALL (F:)
IT Design
JLinkARM_V402d
本地磁盘 (G:)
Recreation (H:)
DVD-RAM 驱动器 (I:)

调试仿真

菜单>Execute

Go：全速运行；
Stop：停止全速运行；
Step in：单步运行，跟踪到被调用函数里边去；
Step：单步运行，把被掉函数当成一整条简单的语句；
Step out：跟踪到函数里面后，可以退出到当前函数的调用处；
Run To Cursor：运行到光标处；
Toggle Breakpoint：对光标所在的行设置或清除断点。



```
;IMPORT  main_entry
AREA    Init,CODE,READONLY

ENTRY

IMPORT  main
BL  main
B   .
END
```

Part 2

用JLINK烧写FLASH

1、配置ADS工程
2、编译生成.bin文件
3、配置jflash工程
4、烧写FLASH

$$\begin{tikzcd}
\boxed{\text{.bin}} \arrow[d] \\
\boxed{\text{flash}}
\end{tikzcd}$$

配置ADS工程

菜单>Edit>DebugRel Settings

需要配置的选项：

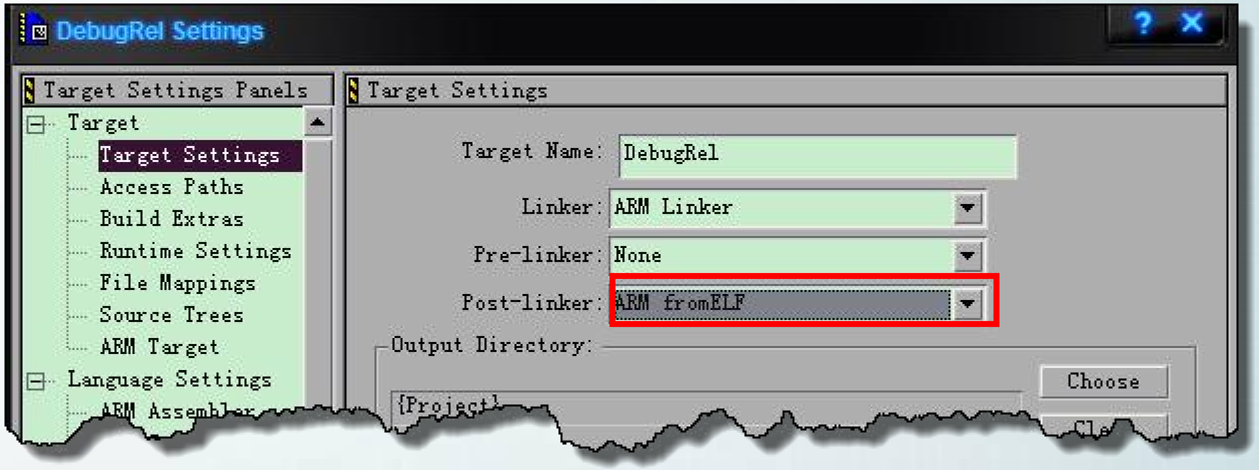
1、Target Settings
2、ARM Assembler（编译器）
3、ARM C Compiler（编译器）
4、ARM Linker（链接器）
5、ARM fromELF

其他选项默认即可。

配置ADS工程

菜单>Edit>DebugRel Settings

1、Target Settings



Target Name: DebugRel

Linker: ARM Linker

Pre-linker: None

Post-linker: ARM fromELF

Output Directory:

{Project}

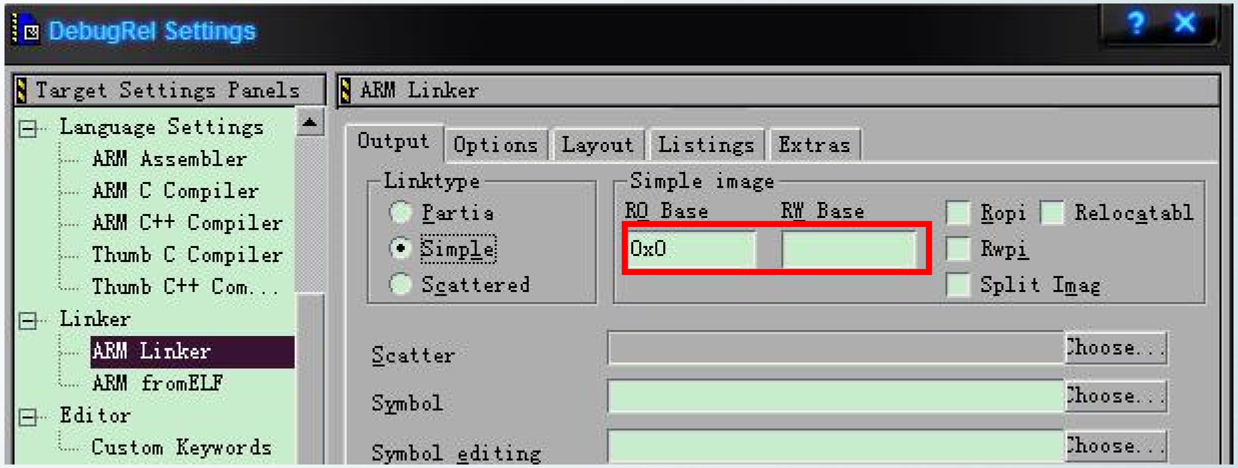
目的：选择链接完成后，对文件进行操作。

链接完成后ADS会默认生成一个.axf的文件，为了得到.bin文件，需要调用ARM fromELF命令将.axf转换为.bin文件。

配置ADS工程

菜单>Edit>DebugRel Settings

4、 ARM Linker（**Output**）


DebugRel Settings

Target Settings Panels
- Language Settings
  - ARM Assembler
  - ARM C Compiler
  - ARM C++ Compiler
  - Thumb C Compiler
  - Thumb C++ Com...
- Linker
  - ARM Linker
  - ARM fromELF
- Editor
  - Custom Keywords

ARM Linker

Output | Options | Layout | Listings | Extras

Linktype: ○ Partia [✓ ● Simple] ○ Scattered

Simple image: RO Base 0x0 RW Base

☐ Ropi ☐ Relocatabl ☐ Rwpi ☐ Split Imag

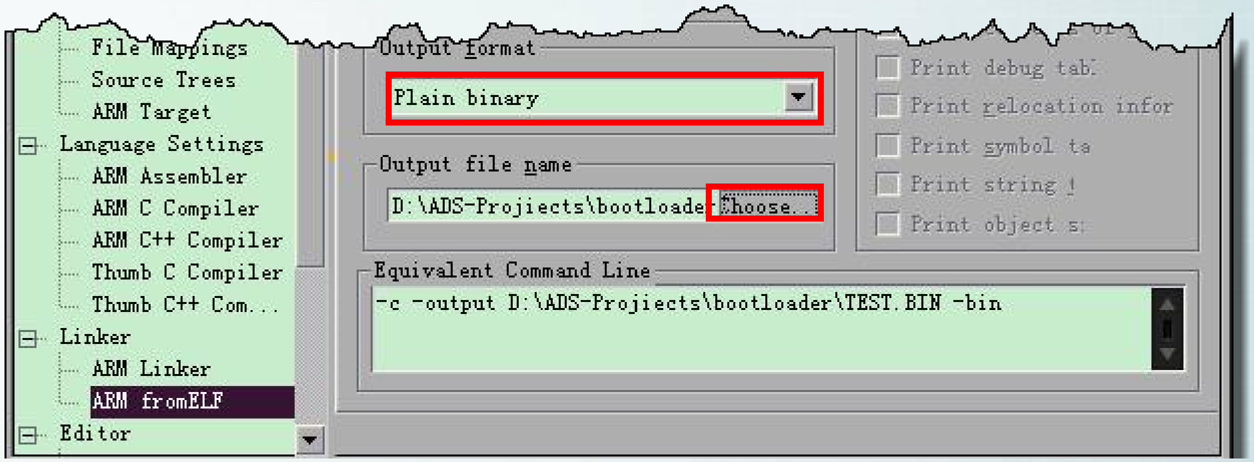
Scatter Choose...
Symbol Choose...
Symbol editing Choose...

目的：设置代码段的起始地址为FLASH的起始地址。

配置ADS工程

菜单>Edit>DebugRel Settings

5、ARM fromELF



Output format: Plain binary

Output file name: D:\ADS-Projiects\bootloade

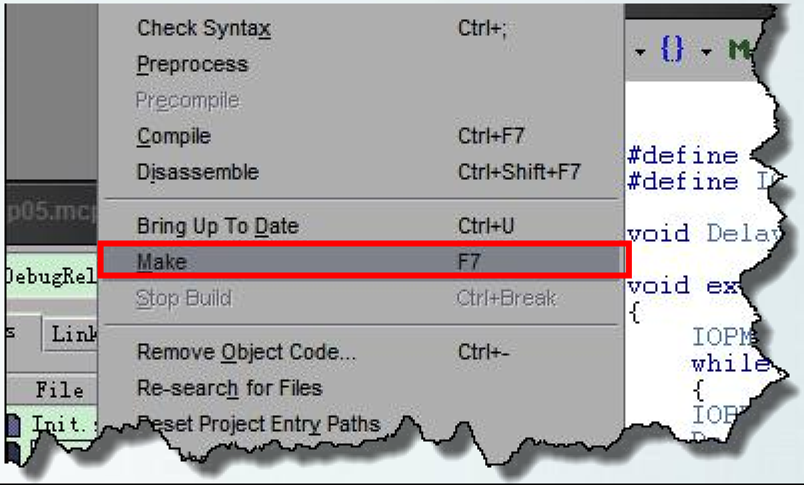
Equivalent Command Line:

```
-c -output D:\ADS-Projiects\bootloader\TEST.BIN -bin
```

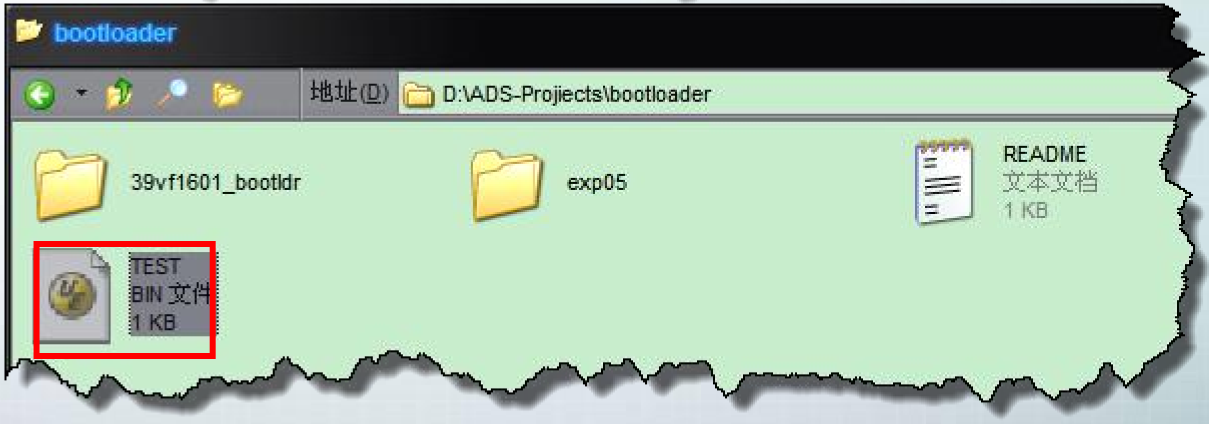
目的：转换为二进制文件并指定输出位置。

编译生成.bin文件

菜单>Projiect>Make



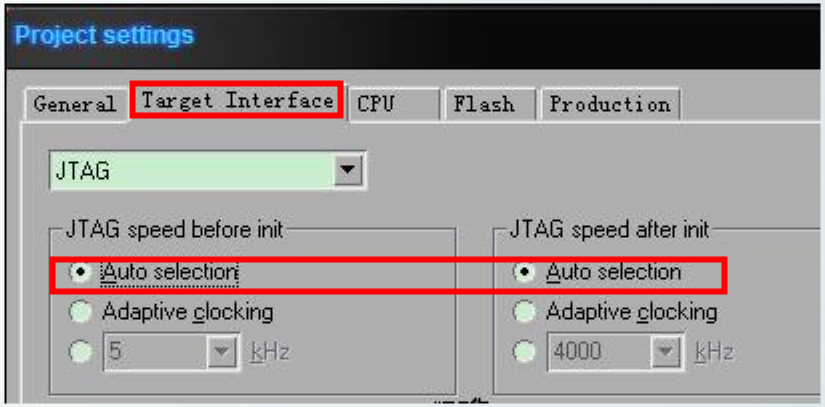
```
#define
#define I
void Delay
void ex
{
    IOPM
    while
    {
    IOP
```



配置jlink工程

菜单>Options>Projiect settings

1、Target Interface



Project settings

General | Target Interface | CPU | Flash | Production

JTAG

JTAG speed before init
- [✓ Auto selection]
- Adaptive clocking
- 5 kHz

JTAG speed after init
- [✓ Auto selection]
- Adaptive clocking
- 4000 kHz

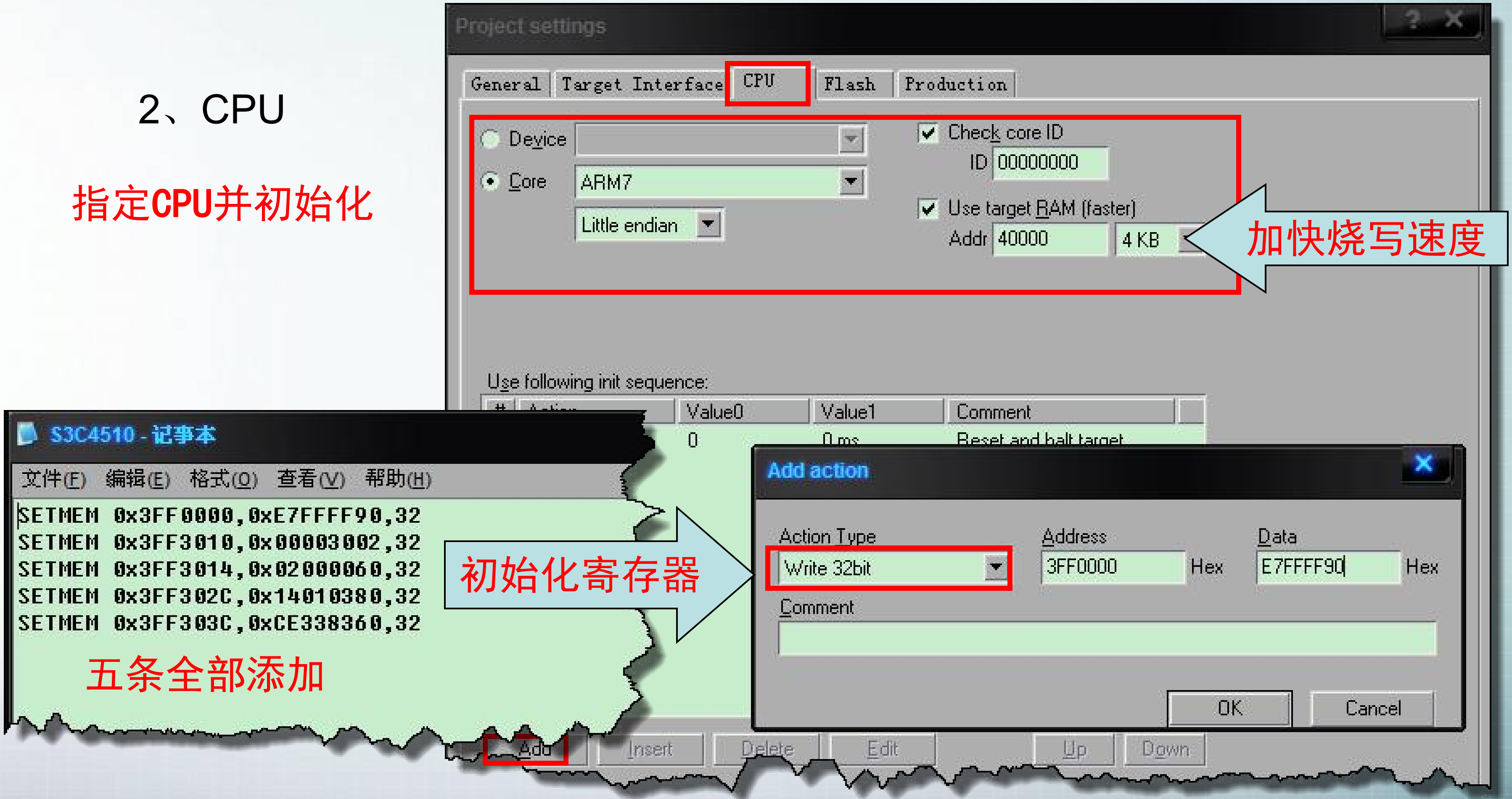
这个根据需要选择，若不确定就选择自动。

配置jlink工程

菜单>Options>Projiect settings

2、CPU

指定CPU并初始化



Device

Core: ARM7

Little endian

☑ Check core ID — ID 00000000

☑ Use target RAM (faster) — Addr 40000, 4 KB

Use following init sequence:

#	Action	Value0	Value1	Comment
		0	0 ms	Reset and halt target

S3C4510 - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
SETMEM 0x3FF0000,0xE7FFFF90,32
SETMEM 0x3FF3010,0x00003002,32
SETMEM 0x3FF3014,0x02000060,32
SETMEM 0x3FF302C,0x14010380,32
SETMEM 0x3FF303C,0xCE338360,32
```

五条全部添加

Add action

Action Type: Write 32bit

Address: 3FF0000 Hex

Data: E7FFFF90 Hex

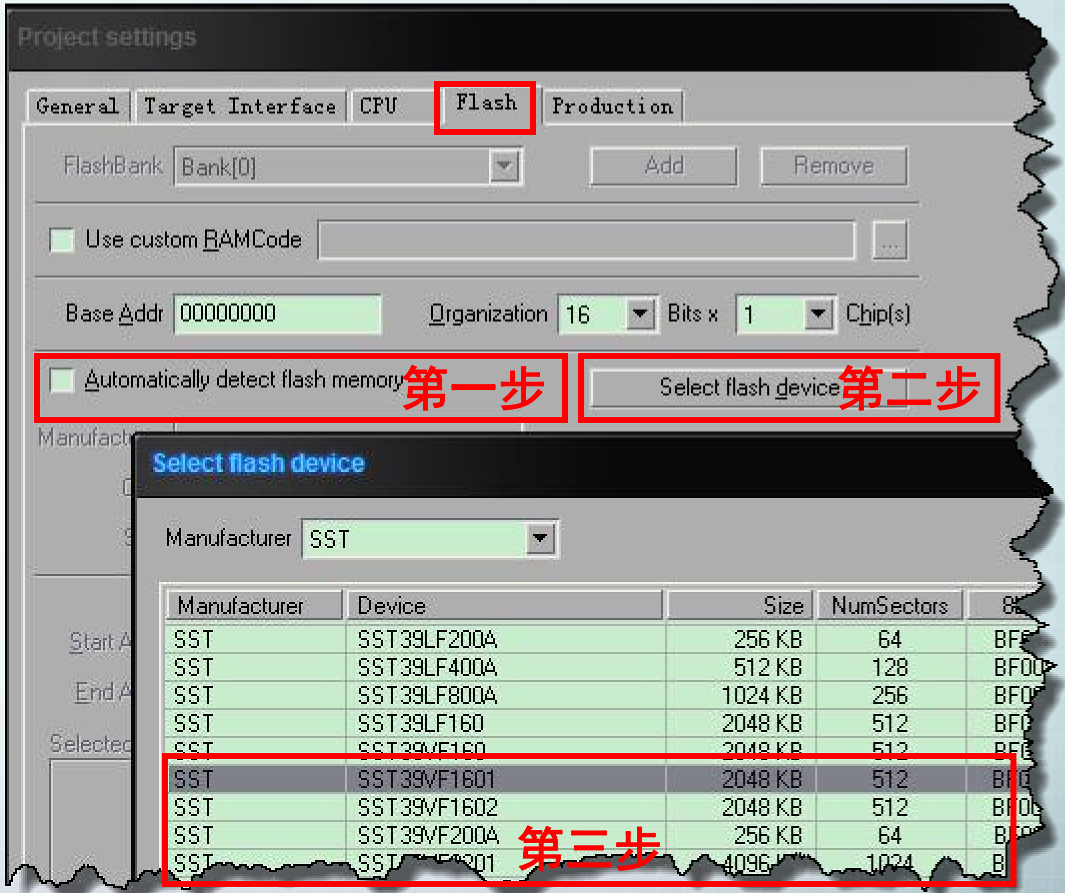
Comment:

配置jlink工程

菜单>Options>Projiect settings

3、FLASH

指定FLASH型号



Project settings

General | Target Interface | CPU | Flash | Production

FlashBank: Bank[0] — Add — Remove

Use custom RAMCode

Base Addr: 00000000 — Organization: 16 — Bits x 1 — Chip(s)

Automatically detect flash memory 第一步

Select flash device 第二步

Select flash device

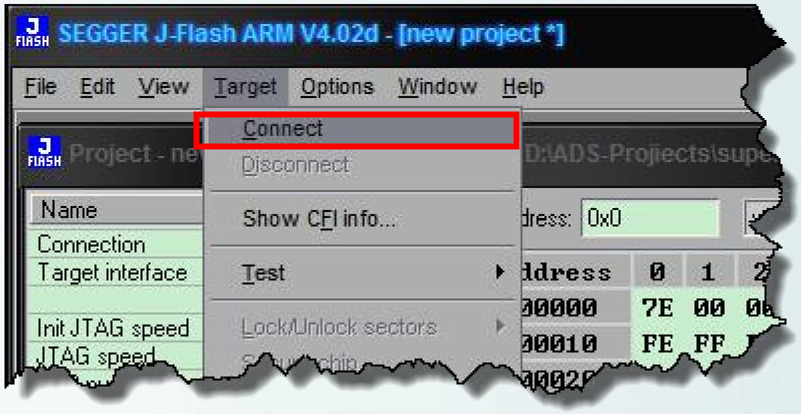
Manufacturer: SST

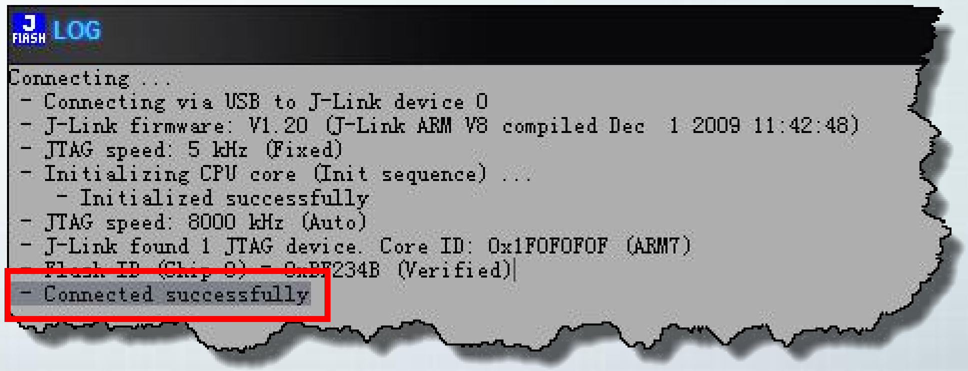
Manufacturer	Device	Size	NumSectors	S
SST	SST39LF200A	256 KB	64	BF
SST	SST39LF400A	512 KB	128	BF00
SST	SST39LF800A	1024 KB	256	BF0
SST	SST39LF160	2048 KB	512	BF
SST	SST39VF160	2048 KB	512	BF
SST	SST39VF1601	2048 KB	512	BF
SST	SST39VF1602	2048 KB	512	BF0
SST	SST39VF200A	256 KB	64	B
SST	SST[illegible]201	4096 [illegible]	1024	B

第三步

烧写FLASH

1、连接实验板

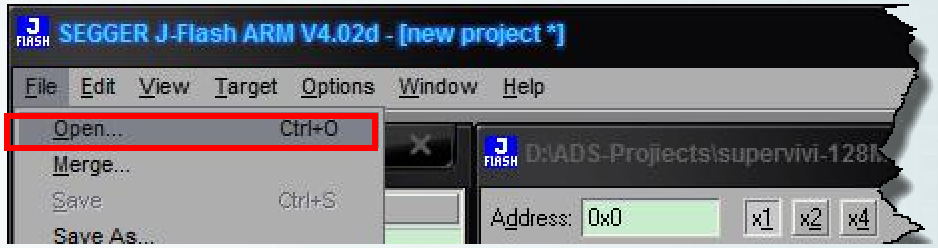


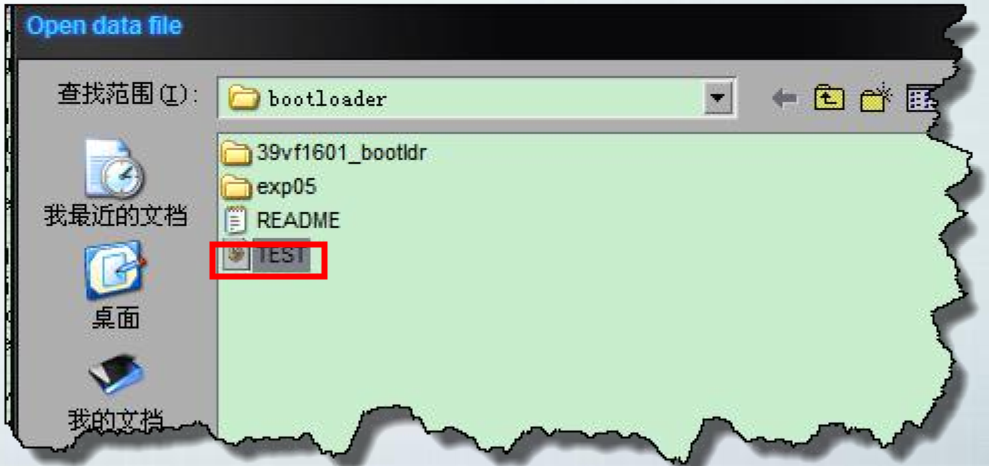

LOG

```
Connecting ...
 - Connecting via USB to J-Link device 0
 - J-Link firmware: V1.20 (J-Link ARM V8 compiled Dec  1 2009 11:42:48)
 - JTAG speed: 5 kHz (Fixed)
 - Initializing CPU core (Init sequence) ...
   - Initialized successfully
 - JTAG speed: 8000 kHz (Auto)
 - J-Link found 1 JTAG device. Core ID: 0x1F0F0F0F (ARM7)
 - Flash ID (Chip 0) = 0xBF234B (Verified)
 - Connected successfully
```

烧写FLASH

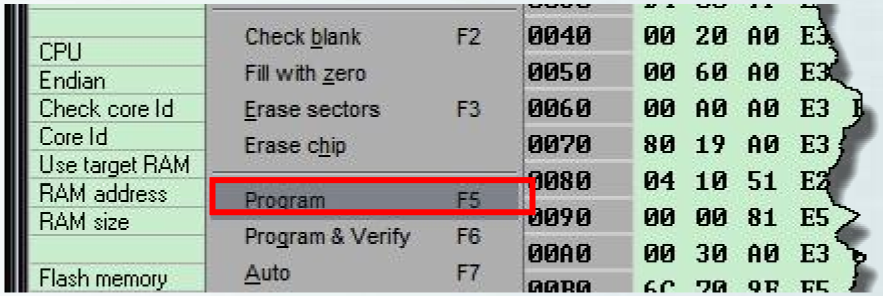
2、加载要烧写的.bin文件


SEGGER J-Flash ARM V4.02d - [new project *]
File Edit View Target Options Window Help
Open... Ctrl+O
Merge...
Save Ctrl+S
Save As...
D:\ADS-Projects\supervivi-128K
Address: 0x0
x1 x2 x4

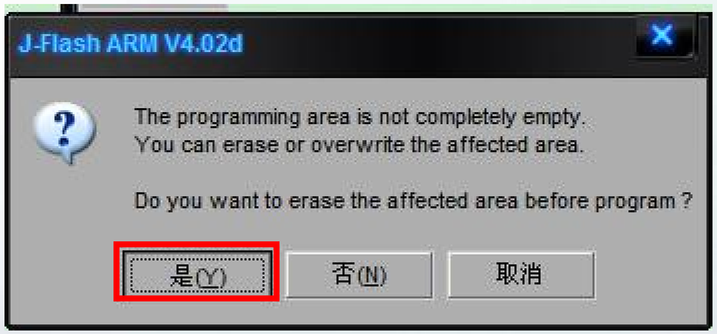


烧写FLASH

3、Program

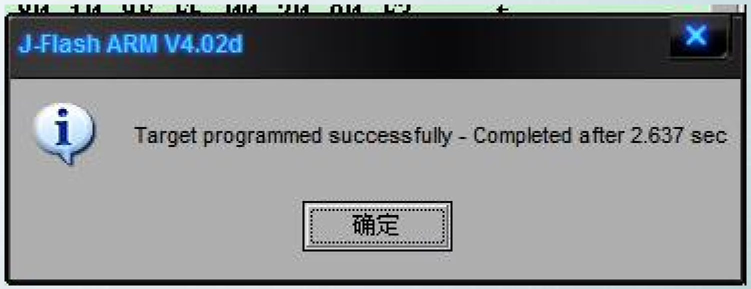


```
0040  00 20 A0 E3
0050  00 60 A0 E3
0060  00 A0 A0 E3
0070  80 19 A0 E3
0080  04 10 51 E2
0090  00 00 81 E5
00A0  00 30 A0 E3
```



The programming area is not completely empty.
You can erase or overwrite the affected area.

Do you want to erase the affected area before program ?



Target programmed successfully - Completed after 2.637 sec

祝大家学习愉快！

Thank You !